TD 6 : Programmation Java
(Les entrée/sorties)

Exercice 1 (La classe File)

Le but de cet exercice est d'utiliser la classe File. Cette classe permet de traiter des fichiers et des répertoires. Elle contient notamment les méthodes `boolean isFile()`, `boolean isDirectory()` et `boolean exists()` qui indiquent respectivement si l'instance correspond à un fichier, un dossier ou un élément existant dans le système de fichiers. Quand l'instance de File correspond à un répertoire, il est possible de lister les fichiers qu'il contient avec `String[] list()`.

1. Créez une méthode `void testFile (String path)` (`path` signifie chemin en français) permettant de tester si le fichier existe, si c'est un fichier ou un répertoire.
2. Si le fichier n'existe pas, alors on le crée en utilisant la méthode `boolean createNewFile()`.
3. Modifiez votre fonction pour les cas où il s'agit d'un répertoire et dans ce cas énumérer l'ensemble des fichiers s'y trouvant en donnant leur chemin absolu avec la méthode `String getAbsolutePath()`.
4. On souhaite à présent afficher que les fichiers dont l'extension est .java. Nous allons pour cela utiliser la fonction `String[] list(FilenameFilter filter)` qui permet de filtrer les listes de fichiers comment on peut le faire avec ***ls***. La Javadoc nous apprend que l'interface `FilenameFilter` décrit une méthode `boolean accept(File dir, String name)` qui doit retourner `true` (donc vrai) si le fichier doit être ajouté à la liste et `false` sinon.

Exercice 2 (Lecture du clavier)

Le but de cet exercice est de calculer la moyenne de notes données au clavier. Le système demande des notes jusqu'à ce que le mot clé « *fini* » soit donné. A ce moment là le système s'arrête et affiche la moyenne. Il faudra faire attention à ce que chaque note soit comprise entre 0 et 20 et soit bien numérique. Pour ceci nous allons utiliser **`System.in`** qui est un **`InputStream`**.
Il est alors possible d'utiliser la classe Scanner permettant la lecture de l'entrée avec :

```
Scanner scanner = new Scanner(System.in)
```

La lecture du clavier se fait alors avec :

```
String line = scanner.nextLine()
```

.

Exercices 3 (Copie de fichiers)

On cherche à écrire un programme effectuant la copie de fichiers. On utilisera pour cela le paquetage `java.io`.

1. Écrire dans un premier temps, une copie de l'entrée standard (`System.in`) sur la sortie standard (`System.out`) octet par octet.
2. Modifier le programme pour prendre deux fichiers sur la ligne de commande si ceux-ci sont spécifiés ; si ceux-ci ne sont pas spécifiés on utilise respectivement l'entrée standard ou la sortie standard.

```
> java Copy fichier.in fichier.out
```

3. Utiliser les entrées/sorties bufférisées (`BufferedInputStream` et `BufferedOutputStream`).

Exercice 4 (Copie des lignes paires)

On cherche à écrire un programme effectuant la copie des lignes paires d'un fichier texte. Le fichier est pris en temps que premier paramètre, la sortie est effectuée sur la sortie standard.

1. En utilisant la méthode `readLine()` de `BufferedReader`.
2. En utilisant un `LineNumberReader`.

Exercice 5 (La commande wc de linux)

Créer une commande qui affiche à l'écran le nombre de lignes, mots et caractères d'un fichier donné en argument.

Exercice 6 (La sérialisation)

Le but de cet exercice est de créer la classe Etudiant. Un étudiant peut se représenter par un nom, un prénom et un ensemble de notes.

1. Créez la classe `Etudiant` avec son constructeur et sa méthode `String toString()` qui affiche son nom et son prénom.
2. On veut pouvoir connaître la moyenne d'un étudiant. Pour cela, nous allons parcourir l'ensemble des notes.
3. On veut aussi pouvoir sauvegarder l'ensemble des éléments.
4. Créez la méthode `static void main(String[] args)` qui devra :

   – demander le nom et le prénom de l'étudiant,
   – demander la liste de ses notes (*réutilisez ce que vous avez fait dans l'exercice 2*),
   – afficher l'instance de l'étudiant et sa moyenne,
   – sauvegarder cette instance dans un fichier ayant pour nom : *<nom de l'étudiant>.etud*.